A Conceptual Graph and RDF(S) Approach for Representing and Querying Document Content

R. Carolina Medina Ramírez, Olivier Corby, and Rose Dieng-Kuntz

Acacia project, INRIA Sophia Antipolis, 2004, route des lucioles
06902 Sophia Antipolis CEDEX, France
{cmedina, corby, dieng}@sophia.inria.fr

Abstract. This article describes a first synthesis of a Conceptual Graph and RDF(S) approach for representing and querying document contents. The framework of this work is the ESCRIRE project [1], the main goal of which is to compare three knowledge representation formalisms (KR): conceptual graphs (CG), descriptions logics (DL), and object-oriented representation languages (OOR) for querying about document contents by relying on ontology-based annotations on document content. This comparison relies on an expressive XML-based pivot language to define the ontology and to represent annotations and queries; it consists of evaluating the capacity of the three KR formalisms for expressing the features of the pivot language. Each feature of the pivot language is translated into each KR formalism, which is than used to draw inferences and to answer queries. Our team was responsible on the CG part. The motivation of this paper is to give a first synthesis of the translation process from the pivot language to RDF(S) and CG, to underline the main problems encountered during this translation.

Keywords: Knowledge representation, Conceptual Graphs, Ontologies, RDFS, XML, and Semantic information retrieval.

1 Introduction

Documents available from the Web or from any digital representation constitute a significant source of knowledge to be represented, handled and queried. The main goal of the ESCRIRE project is to compare three knowledge representation (KR) formalisms in the task of annotating and querying document (by using a specific domain ontology). The test base consist of abstracts of biological articles from the Medline database [2]. This annotated base is queried by each KR formalisms (CG, DL, OOR). A pivot language based on XML syntax was specially defined for the comparison. This pivot language is represented by a DTD containing the syntactic rules to describe an ontology and annotations and to query these annotations using an OQL-based format. Besides, the XML and RDF languages are recommended respectively by the World Wide Web consortium to

F.J. Garijo, J.C. Riquelme, and M. Toro (Eds.): IBERAMIA 2002, LNAI 2527, pp. 121–130, 2002.
© Springer-Verlag Berlin Heidelberg 2002

structure information and to describe any resource on the Web. In [3,4] the impact of using conceptual graphs for indexing and searching information has been studied and analyzed. CG seems to be a good candidate to represent the content of documents. In the following, we would like to stress on the use of conceptual graphs (as an inference mechanism) and on RDF(S) [5,6] language (as an intermediate syntax to represent ontologies and annotations) in the ESCRIRE project. In our experiments we used CORESE, a semantic search engine [7].

The remaining of this paper is structured as follows.In Section 2, we describe briefly the ESCRIRE language. Section 3 presents the translation process from the pivot language to RDF(S) and CGs. In Section 4 we discuss some difficulties encountered during this translation. Section 5 concludes our work.

2 Escrire Language

As explained in the introduction, a *pivot* language was defined (1) to represent a domain ontology, to annotate document contents and to query this base of annotations and (2) to be used as a bridge between the documents constituting the test base and each KR formalism involved in the ESCRIRE project. We recall the main features of this language detailed in [1].

2.1 Description of Domain Ontology

Basically, an ontology is a hierarchy of concepts and relations between these concepts representing a particular application domain. For the ESCRIRE project, a genetic ontology (genes and interactions between these genes) was built and represented by a DTD by one of our colleagues [8]. This DTD defines classes, relation classes, and a subsumption relation organizes these classes into a hierarchy. A class is described by a set of attributes (roles and attributes for the relation classes) the type of which is a class belonging to the ontology (a concrete type, such as integer, string, or Boolean for relation classes attributes). Inside an ontology, a class can be *defined* (i.e. the attribute of the classes are necessary and sufficient conditions for the membership of an individual to this class) or *primitive* (i.e. the attributes of classes are considered as only necessary conditions). For example, the following code represents a relation called *substage*, composed by two roles: SUPERSTAGE and SUBSTAGE the type of which is DEVELOPMENT-STAGE (another class in the ontology). This relation does not have any attribute and it is qualified by properties of transitivity, reflexivity, and antisymmetry.

```
<esc:descbinrel name="substage" transitive="yes" reflexive="yes"
                                antisymmetric="yes">
  <esc:defrole name="superstage">
    <esc:classref name="development-stage"/>
  </esc:defrole>
  <esc:defrole name="substage">
    <esc:classref name="development-stage"/>
  </esc:defrole>
</esc:descbinrel>
```

2.2 Description of Annotations

To each document D_i that belongs to a base of documents corresponds a semantic annotation $Annota_i$ representing the content of D_i. The set of annotations $Annota_1, \ldots, Annota_n$ is represented in the pivot language and uses the conceptual vocabulary specified into the domain ontology, which is also defined in the pivot language. In the ESCRIRE project, an annotation describes *genes* (objects) and *interactions* between genes (relations). The objects are described by attributes (name, class to which they belong to) and the relations are described by *roles* (domain and range) and attributes (to represent the relation characteristics). For example, the annotation given below describes an interaction between a *promoter gene* called *dpp* (with dorso-ventral-system type) and a *target gene*, called *Ubx* (with BX-C type). The effect of the interaction is that the *dpp* gene inhibits the *Ubx* gene.

```
<esc:relation type="interaction">
   <esc:role name="promoter">
      <esc:objref type="dorso-ventral-system" id="dpp"/>
   </esc:role>
   <esc:role name="target">
      <esc:objref type="BX-C" id="Ubx"/>
   </esc:role>
   <esc:attribute name="effect">
      <esc:value>inhibition</esc:value>
   </esc:attribute>
</esc:relation>
```

2.3 Queries

An ESCRIRE query is similar to an OQL [9] query, and is based on the block SELECT (values to be shown in the result), FROM (variables typed by a class), WHERE (constraints to be considered), and ORDERBY (the result order specified by a list of paths). Logical operators (conjunction, disjunction, and negation) as well as quantifiers (universal and existential) can be used. For example, the query given below represents a query to find documents mentioning interactions between the *Ubx* (ultrabithorax) gene acting as target and, as promoter, the *en* (engrailed) gene or the *dpp* (decapentaplegic) gene. The criterion to order answers is the promoter name.

```
SELECT  I.promoter.name, I.effect
FROM    I:interaction
WHERE  (I.target=OBJREF(gene,'Ubx') AND
         ( I.promoter=OBJREF(gene,'en') OR
           I.promoter=OBJREF(gene,'dpp')) )
ORDERBY   I.promoter.name
```

3 Translation Process: Pivot Language-RDF(S)-CG

In this section, we detail the process of translating ontologies and annotations from pivot language to RDF(S) and conceptual graphs. This double translation has two motivations: (1) RDF(S) and pivot language are languages based on XML, and (2) we used CORESE, a semantic search engine based on CG as its inference mechanism and ontology-based annotations represented in RDF(S). Our translation methodology is based on two mappings: (1) *pivot language* → *RDF(S)* - to map from the ESCRIRE ontology and annotations to RDF Schema and RDF statements - and (2) *RDF(S)* → *CG* - to map from an RDF Schema and RDF statements to the CG support and assertional conceptual graphs. A depth comparison and correspondance between CG and RDF(S) models are studied in [10].

To improve clarity, we give a brief overview of the CG model. A conceptual graph [11,12] is an oriented graph that consists of concept nodes and relations nodes describing relations between this concepts. A concept has a *type* (which corresponds to a semantic class) and a *marker* (which corresponds to an instantiation to an individual class). A marker is either the generic marker $*$ corresponding to the existential quantification or an individual marker corresponding to an identifier; M is the individual markers set.

A relation has only a *type*. Concept types and relation types (of same arity) are organized into hierarchies T_C and T_R respectively. This hierarchies are partially ordered by generalization/specialization relation $\geq$ (resp. $\leq$).

A CG support upon which conceptual graphs are constructed is defined as (T_C, T_R, M). The projection operator permits to compare two conceptual graphs, it enables to determine the generalization relation ($\leq$) between two graphs: $G_1 \leq G_2$ iff there exists a projection π from G_2 to G_1. π is a graph morphism such that the label of a node n_1 of G_1 is a specialization of the label of the node n_2 of G_2 with $n_1 = \pi(n_2)$.

In particular, an RDF Schema (the class hierarchy and property hierarchy) corresponds to a CG support (T_C and T_R respectively) and the RDF statements correspond to assertional conceptual graphs in CG.

3.1 Classes and Relations

Since the pivot language and RDF(S) share an XML syntax, we have implemented the mapping *pivot language* → *RDF(S)* as an XSLT style sheet. ESCRIRE ontology is translated into RDF Schema.

To finish the cycle, the mapping *RDF(S)* → *CG* is carried out by CORESE: RDF statements and RDF Schema are translated respectively into assertional conceptual graphs and into CG support.

The definition and description of *classes* and *relation classes* are represented by RDFS classes (rdfs:Class) and in conceptual graphs like concept types belonging to the T_C hierarchy of the CG support. We reified ESCRIRE *relations* classes because, in CG and RDF(S) models the qualified relations (relations with roles and attributes) are not considered i.e., they are translated into RDFS classes

(Concept types in T_C). *Roles* and *attributes* of ESCRIRE relations are translated into RDF properties (rdf:Property); these properties correspond to relation types in T_R. CORESE can handle relation properties like transitivity, reflexivity, symmetry and inverse property. Currently, the antisymmetric property is not handled in CORESE, we are planning to implement it in a future version of CORESE. The binary relations supported by CORESE do not have roles or attributes. We adapted these algorithms for processing ESCRIRE relations (which contain *roles* and *attributes*). The following example shows the CORESE representation of the *substage* relation declare in Section 2.1.

```
<rdfs:Class rdf:ID="substage">
        <cos:reflexive>true</cos:reflexive>
        <cos:transitive>true</cos:transitive>
</rdfs:Class>

<rdf:Property ID="superstage">
        <rdfs:domain rdf:resource="#substage"/>
        <rdfs:range rdf:resource="#development-stage"/>
</rdf:Property>

<rdf:Property ID="substage">
        <rdfs:domain rdf:resource="#substage"/>
        <rdfs:range rdf:resource="#development-stage"/>
</rdf:Property>

<rdfs:Class rdf:ID="development-stage"/>
```

3.2 Objects and Relations

Objects and relations exist inside annotations and in the ontology. The global object features inside the ontology are always true for all the annotations. We consider as global objects: *objects* and *relations* both of them existing in the ontology and referred in the annotations. These global objects represent common and reusable knowledge for annotations, avoiding redundant information. Figure 1, shows two annotations sharing *antenapedia* gene information, located into the global object base.

Objects and relations existing in annotations (ESCRIRE annotations) are represented as RDF statements corresponding to assertional conceptual graphs. The same translation process is applied to global objects.

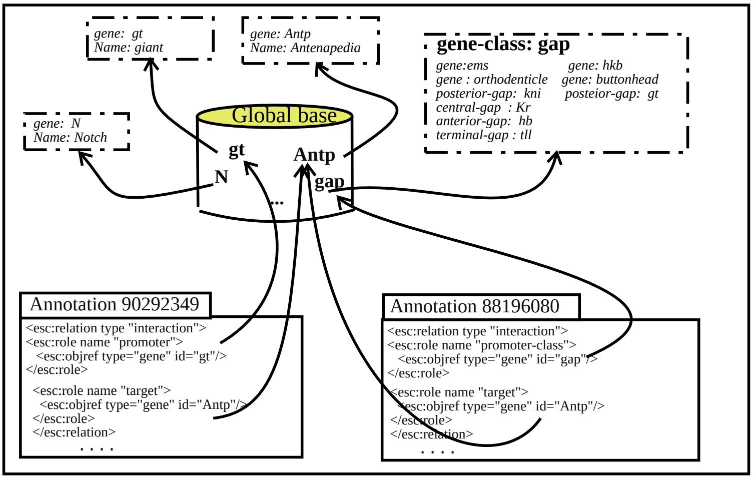
A document is represented by an *individual concept*, for example the concept [Document : `URL-94008526`], is an individual concept with *type field* `Document` belonging to T_C and *marker field* `URL-94008526` belonging to M and representing the URL and document name. The *interaction* relation shown in section 2.2 is represented in RDF and CG in the following way:

Written with RDF syntax:

```
<ns:Interaction>
  <ns:promoter>
     <ns:dorso-ventral-system rdf:about="#dpp"/>
  </ns:promoter>
  <ns:target>
     <ns:BX-C rdf:about="#Ubx"/>
  </ns:target>
  <ns:effect>inhibition</ns:effect>
</ns:interaction>
```

This can be interpreted in CG as:

```
[Interaction : *]->(promoter)->[dorso-ventral-system : dpp]
                 ->(target)->[BX-C : Ubx]
                 ->(effect)->[literal : inhibition]
```



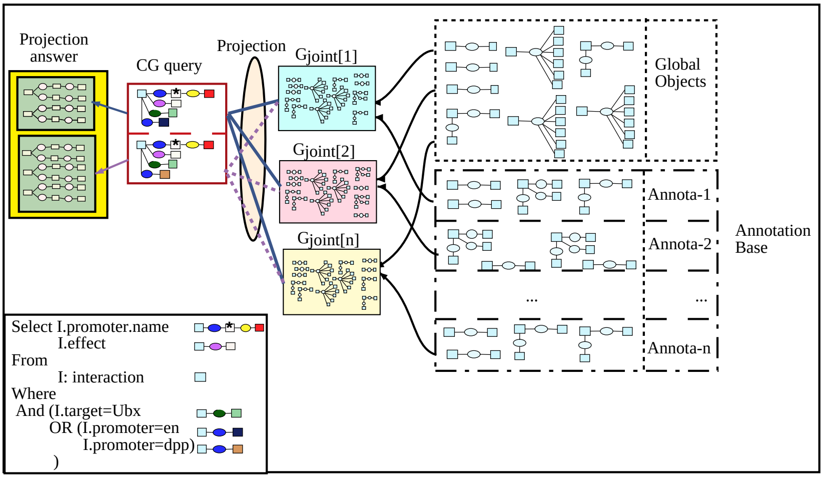
```
<esc:relation type "interaction">
<esc:role name "promoter">
  <esc:objref type="gene" id="gt"/>
</esc:role>
 <esc:role name "target">
   <esc:objref type="gene" id="Antp"/>
 </esc:role>
 </esc:relation>
          . . . .
```

```
<esc:relation type "interaction">
<esc:role name "promoter-class">
  <esc:objref type="gene" id="gap"/>
</esc:role>
<esc:role name "target">
  <esc:objref type="gene" id="Antp"/>
</esc:role>
</esc:relation>
      . . . .
```

Fig. 1. Annotations sharing global object informations.

3.3 Evaluating Queries

Given a document collection, we have an ESCRIRE annnotation that indexes each document (representing its content). These annotations are translated in RDF statements and into assertional conceptual graphs (annotation graphs), constituting an annotation graph base (AG_{Base}). An ESCRIRE query is translated as a query conceptual graph Q and processed by a CG projection of the query Q on the



```
Select I.promoter.name
        I.effect
From
        I: interaction
Where
 And (I.target=Ubx
      OR (I.promoter=en
           I.promoter=dpp)
    )
```

Fig. 2. Query Evaluation using joint graphs.

annotation graph base. The projection operator recovers relevant graphs. Figure 2 shows the process carried out to satisfy a query `Q` considering the global objects. Equation (1) represents the result set retrieved by projection of the query `Q` on the conceptual joint graph (G_{joint}). This joint graph is composed of the global graph (representing the global objects) and each annotation graph staying into AG_{Base}. The annotation graph base size is denoted by N.

$$\begin{aligned} Result = \{\pi\ (Q,\ G_{joint[i]}) \mid \\ G_{joint[i]} = G_{global\ objects} \cup AG_{Base[i]}\}\,, i \in [1, N] \end{aligned} \qquad (1)$$

To compose the joint graph, we select only the relevant arcs of the global graph and the annotation graph. The ϕ operator [11] assigns a first-order formula $\phi(G)$ to each graph G. $\phi(G)$ is a positive, conjunctive and existentially closed formula; therefore, the fact of having the logical disjunction `or`, implies a special processing because disjunction is not considered in the conceptual graphs formalism (traditional projection). We have implemented the `or` operator as follows. A query `Q` is a unique graph if there is not an `or` operator, otherwise `Q` is split into several graphs such that, each of those do not contain `or` operators to be projected on the AG_{Base}. The result corresponding to the query `Q` is composed of the values selected in the query `Q` (select part) and by URL of documents [8]. This result is presented to the user by an XSLT style sheet.

4 Discussion: Translation Problems

In this section we would like to stress on problems faced during the translation from the pivot language to RDF(S) and CG.

Defined classes. The distinction between *defined classes* and *primitive classes* exists in CG but it does not exist in RDF(S). An extension of RDF(S) in order to handle *defined classes* and the *primitive classes* remaining of interest in this case. In CORESE, the translation in CG of the pivot language rests on RDF(S), however none of the mappings *pivot language* → *RDF(S)* or *RDF(S)* → *CG* considers it.

Evaluating relation properties. Binary relations are represented as classes and the properties of transitivity, symmetry, reflexivity and inverse property are processed by a specific code in CORESE. Once again, the RDF(S) model could be extended to support the representation of such kind of properties [13]. In CORESE, several extensions of RDF(S) model have been added for enabling to process this kind of meta-properties and enrich the annotation base. A in-depth description and manipulation of meta-properties are studied in [7].

Handling negation queries. Conceptual graphs correspond to an existential, conjunctive and positive logic, thus the negation is not considered in the formalism of simple conceptual graphs. In some models of conceptual graphs one can put a negation (¬) in some contexts [14]. RDF(S) statements are positive and conjunctive, so handling the negation implies yet another type of extension. In the Notio CG platform [15] on which CORESE was developed, the NOT did not exist. We have implemented particular algorithms to process the negation of each ESCRIRE element. Our main algorithm is the following:
Let be `Q`= $\neg(G_1 \wedge (G_2 \vee G_3))$, where G_1, G_2, G_3 are conceptual graphs.
We have to build :

1. A CG query `Q`' as the normalization of the query `Q` by the application of Morgan's rules.
2. A positive CG query `Q`" (without negation graphs) by replacing negative operators (NOT (A=B)) in each G_i by positive ones (A != B). In this way, only positive graphs are ready to be sent to projection. If there is an `or` operator in `Q`' we split `Q`" such as shown in table 1.
3. The result set (R), as the the projection of each graph in `Q`" on the annotation graph base (AG_{Base}).
4. Validation of R.

The validation process consists of applying the algorithms which process the operator negations (for exemple NOT (A=B)) into the result set R by recalling the original negation criterions.

Evaluating quantifiers. The variables in a query `FROM` clause are existentially quantified. However, the variables in a `WHERE` clause can be quantified. In CG, since the logical interpretation of a graph **G** is a first order logical formula $\phi(G)$, positive and existentially quantified, the existential quantifier can be taken into account. Nevertheless, it is not obvious to handle a universal quantification. A

Table 1. Evaluating negative disjunctive queries.

$\mathtt{Q} = \neg(G_1 \wedge (G_2 \vee G_3))$	Original Query
$\mathtt{Q}' = \neg G_1 \vee (\neg G_2 \wedge \neg G_3)$	Normalization
$\mathtt{Q}'' = G'_1 \vee (G'_2 \wedge G'_3)$ where G'_1, G'_2, G'_3 are positive conceptual graphs $G_4 = (G'_2 \wedge G'_3)$	Positive Query
$R_1 = \pi\ (G'_1, AG_{Base}\ )$ $R_2 = \pi\ (G_4, AG_{Base}\ )$	Projection
$R = R_1 \cup R_2$	Validation

way to support it could be to transform a universally quantified formula into the negation of an existentially quantified formula, but this would require a complex extension of the projection operator.

5 Conclusion

In this paper, we have described a pivot language to represent a domain ontology, annotate document contents and to query this base of annotations. Some expressiveness problems encountered during the mappings: (1)*pivot language* → *RDF(S)* and (2) *RDF(S)* → *CG* have been discussed. These problems underline the need of expressiveness extensions for RDF(S), in order to support defined classes, relation properties, negation and quantifiers. We also described a technique to treat negative disjunctive queries. For this first experiment, a hundred of 4500 abstracts of biological articles extracted from NIH Medline public database have been considered. We have applied a representative query set to validate all aspects of pivot language. Our future efforts are focused on the analysis of semantic expressiveness extension works for RDF(S) [13] and XML [16], to complete our solution. Finally, in order to extend the ontology we are using the inference mechanism provided by CORESE to build a rule base for retrieving hidden knowledge from annotations in order to be exploited afterwards by query evaluations.

References

1. Rim Al-Hulou, Olivier Corby, Rose Dieng-Kuntz, Jérôme Euzenat, Carolina Medina Ramírez, Amedeo Napoli, and Raphaël Troncy. Three knowledge representation formalisms for content-based manipulation of documents. In *Proc. KR 2002 workshop on Formal Ontology, Knowledge Representation and Intelligent Systems for the World Wide Web (SemWeb), Toulouse, France*, 2002.
2. Medline database. http://www.ncbi.nlm.nih.gov/PubMed.
3. Philippe Martin and Peter Eklund. Knowledge Retrieval and the World Wide Web. *IEEE Intelligent Systems*, 15(3):18–25, 2000.

4. Iadh Ounis and Marius Pasca. RELIEF: Combining Expressiveness and Rapidity into a Single System. In *In Proc.of the 21st annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR'98), Melbourne,Australia*, 1998.
5. Resource Description Framework Model and Syntax Specification. W3C Recommendation, 1999.
6. Resource Description Framework Schema Specification. W3C Candidate Recommendation, 2000.
7. Olivier Corby and Catherine Faron. Corese : A corporate semantic web engine. In *Proc. WWW2002 workshop on Real world RDF and semantic web applications, Hawaii*, 2002.
8. Embedded Structured Content Representation In REpositories: Escrire DTD. http://escrire.inrialpes.fr/dtd/.
9. Roderic G. G. Cattell. The Object Database Standard: ODMG-93. Morgan Kauffman, San Francisco, California, 1994.
10. Olivier Corby, Rose Dieng, and Cédric Hebert. A conceptual graph model for W3C Resource Description Framework. In Bernhard. Ganter and Guy W. Mineau(eds.), editors, *In Proc. of the 8th International Conference on Conceptual Structures (ICCS'00), Darmstadt, Germany*, volume 1867 of *LNAI*. Springer-Verlag, 2000.
11. John F. Sowa. *Conceptual structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.
12. John F. Sowa. Conceptual Graphs: DpANS. http://concept.cs.uah.edu/CG/Standard.html, 1999.
13. Alexandre Delteil, Catherine Faron-Zuker, and Rose Dieng-Kuntz. Extension of RDF(S) based on the Conceptual Graph Model. In G.Stumme H.S. Delugach, editor, *Proc. of 9th International Conference on Conceptual Structures (ICCS'01)*, volume 2120 of *LNAI*. Springer-Verlag Berlin Heidelberg, 2001.
14. John F. Sowa. Conceptual Graphs summary. In Timothy E. Nagle, Janice A. Nagle, Laurie L. Gerholz, and Peter W. Eklund, editors, *Conceptual Structures: current research and practice*, pages 3–51. Ellis Horwood, 1992.
15. F. Southey and J. G. Linders. Notio - A Java API for developing CG tools. In William M Tepfenhart and Walling Cyre (eds.), editors, *Proc. of 7th International Conference on Conceptual Structures (ICCS'99)*, volume 1640 of *LNAI*. Springer-Verlag, 1999.
16. W3C Publishes Web Ontology Language Requirements Document. http://xml.coverpages.org/ni2002-03-08-d.html.